Vol. III, No. 3
January/February 1984

satn®

Software Arts Technical Notes from the creators of *VisiCalc*®

CONTENTS	FEATURING	PAGE
Table Talk Chances are good that some of your data is best represented in tables consisting of several rows and several columns. Here we show you how to combine two VisiCalc search functions in order to pull out a value from such a table.	**@CHOOSE** **@LOOKUP**	**2**
From Our Readers A reader offers a tip about recording a worksheet's lower right coordinate so you don't need to search for it when you go to print the sheet.	**/AE**	**5**
Another Look at the VisiCalc Advanced Version Program The VisiCalc program's Advanced Version offers a lot of new features to make your problem solving easier. Here we look at two new built-in functions and some formatting enhancements.	**@LABEL** **@ROUND** **/A** **/F=** **/GCC**	**6**
Book Reviews Barry Bayer looks at two recent additions to the VisiCalc library, *Using VisiCalc: Getting Down to Business*, by Carol Klitzner and Matthew J. Plociak, and *Mastering VisiCalc*, by Douglas Hergert.		**10**
Pointer: Giving Tables a DIFferent Look What do you do if you have a table that goes across a sheet but you want it to go down the sheet? Use the DIF format as demonstrated here.	**/S#**	**11**

TABLE TALK

With the VisiCalc® program's search functions, @LOOKUP and @CHOOSE, you can search for values down one column or across one row. In this article, we show you how to combine two search functions so you can read a table containing many rows and columns. That is, you can specify a row *and* a column in a table, and the corresponding value will be returned.

In the first part of the article, we illustrate the technique with a simple, stripped-down model. In the second part, we create a complete model that contains a pricing table with part numbers and unit prices at different quantities. When the model is completed, you can specify a part number and the quantity ordered, and the model will return the unit price (from the table) as well as the total cost.

The Basic Technique

First, set recalculation to manual and by rows. Type:

```
/GOR
/GRM
```

We want to be able to specify a row and a column in a table and have the model return the corresponding value. We'll set up a purely arbitrary table of numbers from 1 to 20 in cells B4 through E8. Create the table by entering the labels and values through row 10 on the sheet as shown in Figure 1. Row 11 will contain expressions we'll enter below.

Now enter the labels at B13, B14, and B16, as shown in Figure 1. Later we will enter the row and column designators at C13 and C14, respectively.

The VisiCalc program's @CHOOSE function has the syntax *@CHOOSE(v,list)*, where *v* is a number and *list* is a range (such as A1...A10) or a series (such as A1,A2,A3) of cells. The function returns the *v*th element in the specified range or series.

@CHOOSE deals with only one row or column at a time. So we will use two @CHOOSE expressions. One will use the row designator we give at C13 to extract the appropriate row from the table, and the second will use the column designator given at C14 to pick out the proper value from the extracted row.

To do this, we need to create a place for the appropriate row to be picked out of the table. We'll do this in row 11 on the sheet. Enter:

```
>B11:@CHOOSE(C13,B4...B8)
```

That expression takes the row designator (to be given at C13) and returns the value in the specified column (B4...B8) at the given row. Here, it returns @NA. Since cell C13 is currently blank, the VisiCalc program reads its value as 0; there is no "zeroth" element in a series, so the expression returns @NA.

How to Use SATN

The VisiCalc program is available on many different personal computers. These machines may have slightly different keyboards. To reduce confusion, SATN will use the symbol ® and the word RETURN synonymously to represent the keys known as ENTER, ENDLINE, and RETURN. The symbol ^ is used to represent the exponentiation operator.

To make VisiCalc documentation more readable, the program was designed so that the colon (:) has the same effect as RETURN in some commands, as when ending the > command and to separate the various elements of the replicate command.

In examples of what to type, type exactly what you see and press RETURN at the end of every line. On some computers, you may find it necessary to add parentheses to some of the expressions included in the text. For example, 3 + 4*5 may become (3 + 4)*5.

Throughout SATN, one typeface represents both VisiCalc input and output (what you should type and what you will see on the screen). The typeface looks like this:

```
This line is typeset in OCR-B.
So is this: +A1+B1*3.25
```

Characters you should type in are printed in rust.

To allow you to type the formulas directly into VisiCalc, the examples use specific coordinates, like A1, rather than the names of the value they represent. When actually typing a formula, you can use the arrows to point rather than using the > command.

SATN is published bimonthly by Software Arts Products Corp., 27 Mica Lane, Wellesley, MA 02181. Address all editorial correspondence to the Editorial Department, SATN, P.O. Box 99, Newton, MA 02162. Requests for special permission or bulk orders should be addressed to the publisher.

Domestic subscriptions: 6 issues $30.00, 12 issues $50.00; outside U.S.A., 6 issues $50.00, 12 issues $90.00. Single-copy price, $6.00. Back-issue price, $7.00. Send subscriptions, change of address, and fulfillment questions to SATN Subscriptions, P.O. Box 97, Newton, MA 02162.

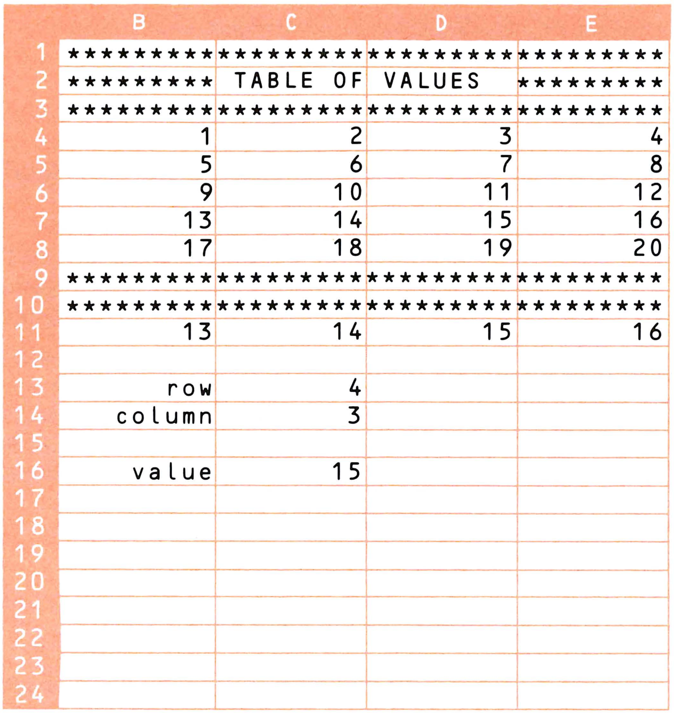
Copyright © 1983, Software Arts Products Corp. All rights reserved. This periodical may not be copied or reproduced in any fashion either in whole or in part without the prior written consent of Software Arts Products Corp.

Software Arts Products Corp. makes no warranty, express or implied, with respect to the quality, accuracy, or freedom from error of the information contained in SATN, the journal for VisiCalc users, and specifically disclaims all liability for any damages resulting from the use of such information.

Unless specifically stated otherwise in any article in this issue of SATN, Software Arts Products Corp. has not examined any products and/or publications identified in SATN to determine their accuracy and/or ability to perform as advertised. Mention of a product and/or publication in SATN does not constitute an endorsement by Software Arts Products Corp. of any such product or publication.

Software Arts is a trademark of Software Arts, Inc. and Software Arts Products Corp. Unless specifically stated otherwise, all references in this issue of SATN to Software Arts shall refer to both companies. DIF is a trademark of Software Arts Products Corp. Keystroke Memory is a trademark of Software Arts Products Corp. SATN is a registered trademark of Software Arts Products Corp. VisiCalc is a registered trademark of VisiCorp.

Apple is a registered trademark of Apple Computer, Inc. IBM is a registered trademark of International Business Machines, Inc. Atari is a trademark of Atari, Inc. Commodore is a trademark of Commodore International. HP-83 and HP-85 are trademarks of Hewlett-Packard Company. TRS-80 is a trademark of Radio Shack division of Tandy Corp.



	B	C	D	E
1	*********	*********	*********	*********
2	*********	TABLE OF	VALUES	*********
3	*********	*********	*********	*********
4	1	2	3	4
5	5	6	7	8
6	9	10	11	12
7	13	14	15	16
8	17	18	19	20
9	*********	*********	*********	*********
10	*********	*********	*********	*********
11	13	14	15	16
12				
13	row	4		
14	column	3		
15				
16	value	15		
17				
18				
19				
20				
21				
22				
23				
24				

FIGURE 1. *This model uses two @CHOOSE expressions to pick values out of a table consisting of multiple rows and columns. The first @CHOOSE extracts the appropriate row from the table, based on the value given in cell C13. The second @CHOOSE selects the appropriate value from the extracted row (found in cells B11...E11), given the column specification at C14.*

Replicate the expression at B11 across to E11 by entering:

```
>B11:/R:C11...E11:NRR
```

All the formulas on row 11 evaluate to @NA.

For a row designator, enter:

```
>C13:4
```

Type ! to calculate the sheet. All the fourth-row values from the table are now repeated on row 11 on the sheet.

Now that we have extracted the correct row, we can use a second @CHOOSE expression to pick out the value in the proper column. Enter:

```
>C16:@CHOOSE(C14,B11...E11)
```

That expression says to take the value in the correct column (to be specified in C14) from the isolated range (B11...E11).

Enter a column specifier:

```
>C14:3
```

Type !. The value 15 is returned, indicating the value in the table's fourth row, third column.

Change the row designation to 2 and the column designation to 4. Now type !. The values at row 11 on the sheet now duplicate the second row in the table of values. At C16, 8 is returned, because 8 is the value in the second row, fourth column of the table.

We have used one @CHOOSE expression to search values found by another @CHOOSE. Some points to note about this technique:

- We could have had the first @CHOOSE expression pull out a *column* instead of a row. In this case, the second @CHOOSE would then use the row designator to extract the desired value.
- You can also use @LOOKUP as a search function, but that is a bit more awkward since you need to set up an extra column or row to serve as the index range.
- You can hide the extracted row (or column) with the graph format (/F*).

A More Useful Model

The above model illustrates the basic technique but has limited utility; you have to specify the row and column manually. It would be better if the model could determine where in the table the desired value is. We'll now create a model that can.

We'll use a table of product information that includes part numbers and unit prices at different quantities. When we have finished the model, we will be able to specify the part number (P/N) and quantity ordered. The model will determine where in the table the unit price is, return the unit price, and calculate total cost.

Clear the sheet by typing /CY. Specify manual recalculation by rows and right-aligned format by typing:

```
/GRM
/GOR
/GFR
```

Enter the row and column headings as follows:

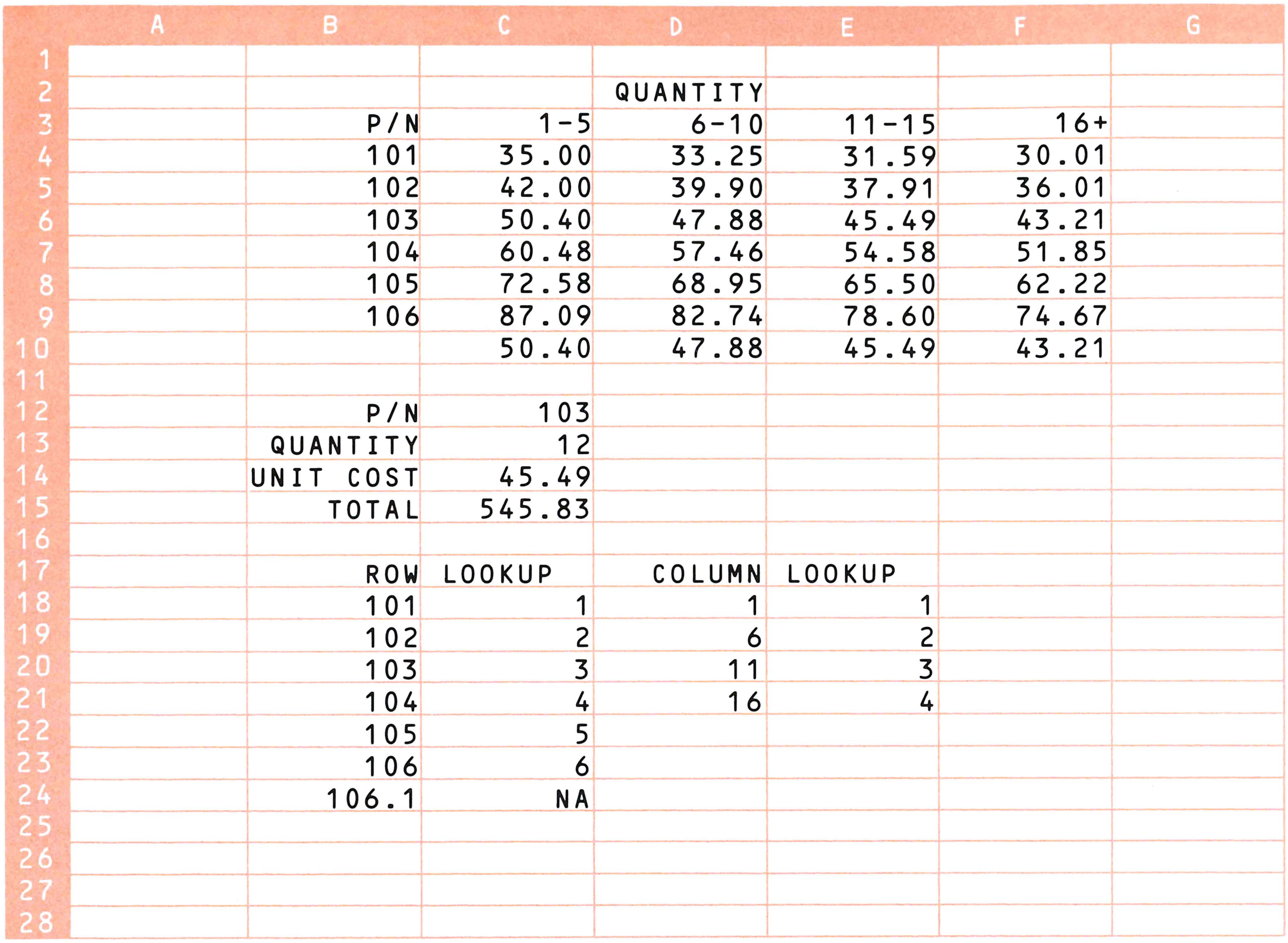
```
>D2:QUANTITY
>B3:P/N
>C3:"1-5
>D3:"6-10
>E3:"11-15
>F3:"16+
>B4:101
>B5:1+B4®/R:B6...B9:R
```

Let's assume that at each step up in quantity ordered the unit price drops 5 percent and that each part is 20 percent more expensive than the previous one in the list. Enter these values:

```
>C4:/F$35
>C5:/F$1.2*C4®/R:C6...C9:R
>D4:/F$.95*C4®/R:E4...F4:R
>D4:/R...F4:D5...D9:RRR
```

We will enter the part number and quantity ordered at C12 and C13, respectively. The unit price and total cost will be returned in C14 and C15, respectively. Enter the corresponding labels in column B in rows 12 through 15 as shown in Figure 2.

To serve as examples, enter these values:



	A	B	C	D	E	F	G
1							
2				QUANTITY			
3		P/N	1-5	6-10	11-15	16+	
4		101	35.00	33.25	31.59	30.01	
5		102	42.00	39.90	37.91	36.01	
6		103	50.40	47.88	45.49	43.21	
7		104	60.48	57.46	54.58	51.85	
8		105	72.58	68.95	65.50	62.22	
9		106	87.09	82.74	78.60	74.67	
10			50.40	47.88	45.49	43.21	
11							
12		P/N	103				
13		QUANTITY	12				
14		UNIT COST	45.49				
15		TOTAL	545.83				
16							
17		ROW	LOOKUP	COLUMN	LOOKUP		
18		101	1	1	1		
19		102	2	6	2		
20		103	3	11	3		
21		104	4	16	4		
22		105	5				
23		106	6				
24		106.1	NA				
25							
26							
27							
28							

FIGURE 2. *This model expands on the technique illustrated in Figure 1. It uses @LOOKUP tables to translate the known information (an item's part number and quantity) into row and column references. Two @CHOOSE functions operate on those references to pick the proper unit cost from the table of values found in rows 4 through 9.*

We must provide the mechanism that translates the part number and quantity into a row and column designator so that the table can be read. We will do that with two short @LOOKUP tables.

Looking Up the Row and Column

An @LOOKUP table consists of a range of values to be searched and their corresponding values. The @LOOKUP function, which has the syntax *@LOOKUP(v,range)*, takes the value *v* and searches the specified range until it reaches the first number in the range that is greater than *v*. It then returns to the preceding number in the range and selects the corresponding value from the column or row to the right of or below the entries in the range. It assumes the values in *range* are in ascending order.

Our first @LOOKUP table will relate the part number to the corresponding row in the table of values.

In columns B and C, starting at row 17, enter the headings and values for the row @LOOKUP table, as shown in Figure 2. Using this table, we can search the part numbers (101 through 106) and have the corresponding row number determined (1 through 6). Notice that the entry at B24 (106.1) returns @NA. That serves as an error trap; if a part number greater than 106.1 is given, @NA is returned, indicating that there is no such part number.

The second @LOOKUP table will pick out the proper column from the table of values. Notice that each column applies to a range of units (1 through 5 items, 6 through 10, and so on). We will use the crossover points (6, 11, and 16) as the @LOOKUP values.

Enter the labels at D17 and E17 and the values in columns D and E, as shown in Figure 2.

Using this @LOOKUP table, when a quantity is specified, the corresponding column is returned. Any number less than 6 will return the first column, any number between 6 and 10 will return the second column, and so on.

Now we are ready to enter the expressions that will pull the appropriate value out of the table in rows 4 through 9. As in the first example in this article, we will extract the relevant row from the table using the first @CHOOSE expression. However, instead of simply entering the expression as *@CHOOSE(row designator, range)*, as we did above, we will embed an @LOOKUP expression that will return the row designator. Enter this expression:

```
>C10:/F$@CHOOSE(@LOOKUP(C12,
                 B18...B24),C4...C9)
```

Replicate that expression across row 10 by entering:

```
>C10:/R:D10...F10:NNNRR
```

The row in the table corresponding to the part number indicated at C12 is extracted here. In this case, the third row is pulled out.

Now all that's left is to use a second @CHOOSE to select the correct value from the extracted row. The expression is similar to the one just entered; it uses an @LOOKUP function to determine the column, then looks up the column in the extracted row. Enter:

```
>C14:/F$@CHOOSE(@LOOKUP(C13,
               D18...D21),C10...F10)
```

Enter the expression to calculate the total cost:

```
>C15:/F$+C13*C14
```

Now the model is complete. The unit cost for 12 of part number 103 is $45.49; the total cost is $543.83.

Change the desired part number and quantity by entering:

```
>C12:105
>C13:7
```

Recalculate the sheet. The unit cost for 7 items of part number 105 is $68.95; the total cost is $482.63.

Conclusion

The table of values can be expanded easily to any desired size. You only have to enlarge the @LOOKUP ranges and @LOOKUP tables. You can have the table of values anywhere on the sheet. Since it contains values that don't change, forward references are not a problem. You should, however, keep the extracted row above where you want the unit cost and total cost to be calculated to avoid a forward reference and the necessity of recalculating the entire sheet.

We used two @LOOKUP tables to translate the known information (part number and quantity) into row and column designators. In the example given, we could have simply subtracted 100 from the part number to get the row, since 101 corresponds to the first row, 102 to the second, and so on. But that was only coincidental. If there had been gaps in part numbers (for example, no part number 104), the expression would not return the desired row; an @LOOKUP table will always do the trick. An @LOOKUP table also has the virtue of being able to determine the appropriate row or column when each applies over a range of values, as demonstrated in the above example where each column in the table applied to several quantities.

FROM OUR READERS

I have a hint that I would like to share with your readers.

I find it tedious to search for the lower right corner of my model before printing. So I start my worksheet at A2 and reserve A1 for the coordinate of the lower right corner. I use VisiCalc® Advanced Version, so, for instance, if the lower right corner of what I want printed were at K35, I would enter:

```
>A1:+K35
/AEY
```

This causes the row number and column letter to appear at A1. I can add or delete a row or column in the body of the worksheet (except row 1 or column A) and the coordinate in A1 is automatically updated. Also, I often make other notes to myself in row 1.

Pat Grappe
Levelland, Texas

Editor's Note: To use Pat's suggestion, you needn't have VisiCalc Advanced Version. In other versions of the program, if your worksheet were to end at K35, you would simply enter:

```
>A1:+K35
```

When you are ready to print your model, return to A1 and you can read the lower right coordinate on the status line. In fact, the same thing will happen in the Advanced Version even without setting the attribute to display the expression (/AEY). By setting the attribute in the Advanced Version, you ensure that the lower right coordinate always appears in cell A1, as well as on the status line.

Remember that you can reserve a row to record a coordinate or make other notes to yourself at any location on your worksheet. For example, if the printed portion of your model begins at G20, you can reserve row 19 starting at column G to enter notes that may be useful to you.

ANOTHER LOOK AT THE VISICALC ADVANCED VERSION PROGRAM

Among the features added to the Advanced Version of the VisiCalc® program are the @LABEL and @ROUND functions and user-defined formats. In this article we'll demonstrate these new features in a hypothetical example involving decisions to buy or sell stocks based on quarterly earnings. Before you conclude that we are about to share some insights into stock selection using the VisiCalc program, we should point out that we are not in the business of rendering investment advice and that the stock selection criterion we will use in our example is intentionally naive and downright unrealistic. We picked it to show how to use the program. So if you are in the markets, bear with us and you'll learn more about the VisiCalc program, but you'll still have to look elsewhere for real investment advice.

Formatting the Sheet

First set the global column width to 10 characters with the entry:

```
/GC10
```

Now, set column A to four characters. Enter:

```
>A1:/GCC4
```

Before entering any values, let's set the global display of values to show negative values within parentheses, commas, dollar signs, and integers.

In the VisiCalc Advanced Version program, you can define a new format or change the definition of a predefined format (such as /FI, /FR, /FL, or /F$) by using /F= and then giving the name of the format. Once you enter /F= and give the format a name, the program presents you with the phrase "User Format:" and a short alphabetic menu that is essentially the same as the one used in Attribute mode. (There's only one difference: When using /F=, the menu includes the option O, which can be used to return a format to its original setting.) In short, once you have named a format, you enter Attribute mode. So, to redefine /F$, type:

```
/F=$V(,$I
```

The prompt line should read "V(,$I:". Notice that when you named the format $ and specified V for value, the term 2F, which was the entire pre-existing definition of /F$, was part of the information that appeared on the edit line. When you entered I, the term 2F disappeared, since the two are exclusive choices. It's also worth pointing out that the order in which you enter the format definitions does not matter.

Enter the redefined format.

Now enter:

```
/GF$
```

If you were now to enter 765432 anywhere on the sheet it would display as $765,432. Similarly, .45 and −432.76 would display as $0 and ($433). Cells containing local formats, however, will not display their contents with the global format.

We want all labels to be centered, so enter:

```
/GALC
```

While we're redefining formats, let's also redefine /FI to include commas in values and to center labels. Enter:

```
/F=IV,®LC
```

Now, whenever /FI is entered in a cell, any value over 1000 that appears there will display a comma; any label entered in the cell will be centered. (For more information on user-defined formats, see SATN Vol. II, No. 3, January/February 1983.)

Entering the Values

We have set up the sheet shown in Figure 3 with three labels, BUY, SELL, and WAIT, in A1...A3. Enter those labels as shown in the figure.

In B1 and B2, we have associated values with the labels BUY and SELL. Format cells B1 and B2 with /FI and enter `1` and `0` as shown in the figure.

Now use the Repeating Label command (/-) to create dashed lines in rows 7 and 9 and enter the labels shown above the top line.

We've put this company's Quarterly Earnings per share on a bit of a roller coaster so that we can easily shift from BUY to SELL merely by changing a single value. Notice that the earnings shown in Figure 3 are formatted to show dollar signs and two decimal places. To override the global settings that are in effect, we will set local attributes in column B and replicate them into columns C through H. So, before entering the values shown in the figure, enter:

```
>B8:/AV$2F®®/R:C8...H8
```

Now if you scroll through cells C8...H8, you will see that the attributes you set in B8 control the display in those cells, too. Had you already entered a value in B8, though, you could still have replicated only the attributes into cells C8...H8 by entering /R(A) instead of /R in the expression above. Now enter the Quarterly Earnings shown in the figure in cells B8...H8.

Notice that column F is empty at this stage. We'll use it later for a series of expressions. Now enter the label Current Price, as shown in the figure. Since stock prices are often quoted in multiples of eighths, we will use 2-5/8 as the price. That means it's necessary to set local attributes at D10. Enter:

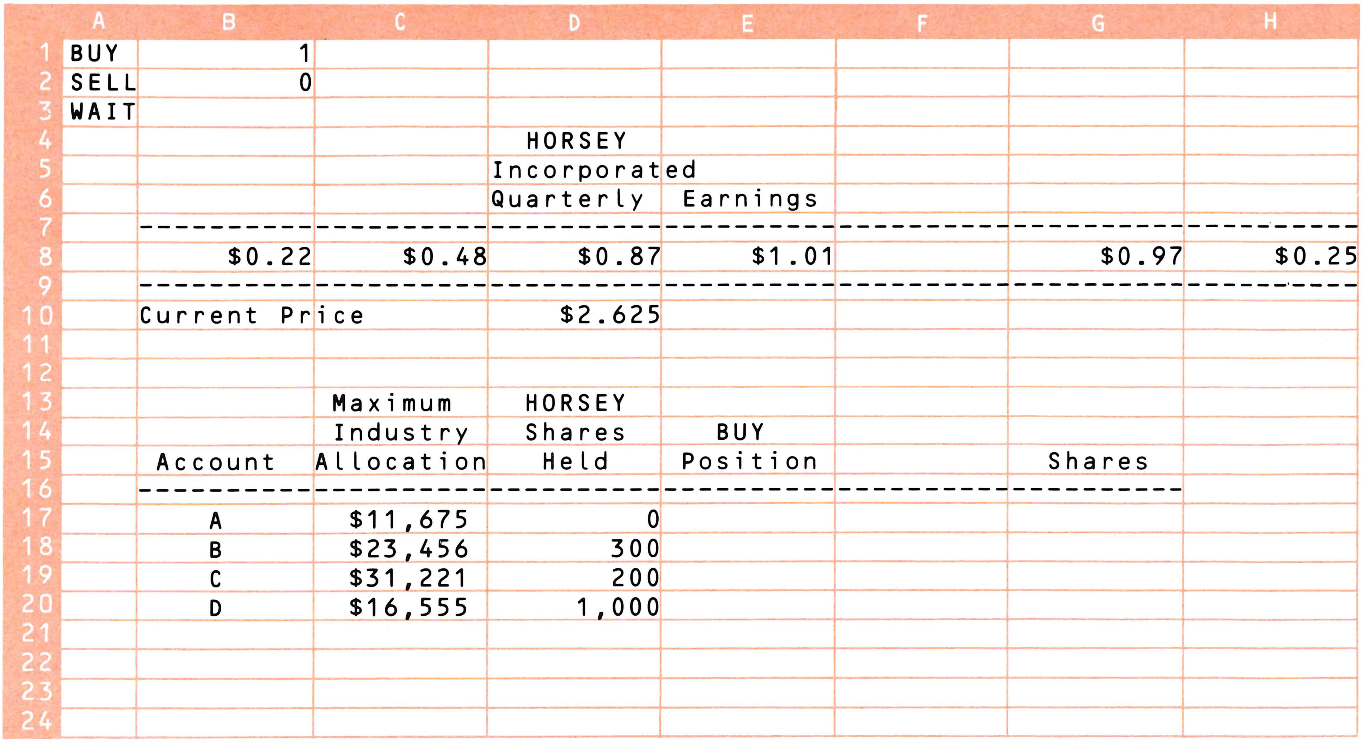
```
>D10:/AV$3F
```

That sets the display to show a dollar sign and three decimal places. Now enter `2+(5/8)`; it displays as $2.625.

Using @LABEL

The Quarterly Earnings that we've entered above are those of HORSEY Incorporated, a hypothetical manufacturer of harnesses and hames. Because we anticipate that we'll look at other companies' prospects too, at D13 we've entered the first @LABEL. Enter:

```
>D13:@LABEL(D4)
```



	A	B	C	D	E	F	G	H
1	BUY	1						
2	SELL	0						
3	WAIT							
4				HORSEY				
5				Incorporat	ed			
6				Quarterly	Earnings			
7		----------	----------	----------	----------	----------	----------	----------
8		$0.22	$0.48	$0.87	$1.01		$0.97	$0.25
9		----------	----------	----------	----------	----------	----------	----------
10		Current Pr	ice	$2.625				
11								
12								
13			Maximum	HORSEY				
14			Industry	Shares	BUY			
15		Account	Allocation	Held	Position		Shares	
16		----------	----------	----------	----------	----------	----------	
17		A	$11,675	0				
18		B	$23,456	300				
19		C	$31,221	200				
20		D	$16,555	1,000				
21								
22								
23								
24								

FIGURE 3. *In this and the other figures accompanying this article the labels that are used or returned by the @LABEL function are all in uppercase letters. All the other labels contain lowercase letters. "Current Price" and "Incorporated" are locally formatted to override the global format that centers all labels.*

Now, by changing the label at D4 from HORSEY to UPENDOWN, or any other character string, you can see a simple use of the @LABEL function. But notice that if you enter a value at D4 that value will appear at D13 as well. The @LABEL function returns whatever is in a referenced cell.

Enter the other labels shown in rows 13...16, but do not make an entry at E14. Cell E14 will contain a statement that returns the label in A1 or A2.

Before dealing with that expression, though, enter the account names and Maximum Industry Allocation shown for each account in B17 through C20. The values for Maximum Industry Allocation have been plucked from the air and represent the most that any account can have invested in harness and hame manufacturers. If this were a real situation, taken from a portfolio manager's worksheet, such values might be calculated from many other values. Here they are entered as constants. Also enter the number of shares of HORSEY Incorporated held for each account; format them locally with /FI.

Now return to cell E14 to enter the @IF statement that returns the label BUY or SELL. We've made it a simple (even simplistic) matter of calculating and comparing the average earnings of the first four-quarter period and the last four-quarter period. Since only six quarters are shown, there's an overlap of two quarters. If the average of quarters one through four is less than the average of three through six, then the statement returns the label at A1 (BUY); otherwise, the label at A2 (SELL) is returned. (Remember that column F is empty along row 8, meaning that the first @AVERAGE reads four cells, while the second reads five. Nonetheless, both @AVERAGEs actually evaluate four quarters, since @AVERAGE does not include blank cells in its count.) Enter:

```
>E14:@IF(@AVERAGE(B8...E8)<@AVERAGE
     (D8...H8),@LABEL(A1),@LABEL(A2))
```

With the Quarterly Earnings shown in Figure 3, E14 displays BUY. To see the program change BUY to SELL, change B8 from .22 to 1. Before continuing, change it back to .22.

Overall, though, should we buy HORSEY stock? This oversimplified analysis says to do that, but what about individual accounts? Before racing out to buy shares of HORSEY we want to be sure that at the current price we aren't already holding too many shares. Let's assume that we should not risk more than 5% of an account's allocation in any single stock in the harness industry. Thus, we have to test whether, at the current price, the shares already held for an account are equal to or less than 5% of the Maximum Industry Allocation. If so, we want to buy. If not, we want to sell. Enter:

```
>E17:@IF(D10*D17<=(C17*.05),
              @LABEL(A1),@LABEL(A2))
```

The term D10*D17 produces the dollar value of the shares already held, while C17*.05 gives us the dollar value of 5% of the Maximum Industry Allocation for the account. In this case no shares are held for the account, so D10*D17 is 0, definitely less than 5% of $11,675, and the expression returns BUY, the label at A1.

Now replicate E17 into the other accounts. Enter:

```
>E17:/R:E18...E20:NRRNN
```

That produces BUY, BUY, BUY, and SELL in E17...E20.

Were we inclined to make investment decisions lightly, we might stop at this point and trade shares of HORSEY. But there are two problems left to resolve. First, given the number of shares held for each account, how many should we buy or sell? Second, we don't want to deal in odd lot trading, so we have to round the number of shares traded to even hundreds.

The first problem, deciding how many shares, is more difficult than the second. Notice that the @LABEL function will return whatever is in a cell. If, for example, A1 or A2 were blank, E14 would also be blank. If they contained values, then E14 would also contain a value. At this stage, column E contains labels that tell us whether to buy or sell, but the VisiCalc program cannot distinguish between the labels BUY and SELL; all labels, from the program's point of view, have a value of zero. However, to make any further decisions, we need to distinguish between the two labels. That's why, at the beginning of this exercise, we associated BUY with the value 1 and SELL with 0. By so doing, we can simply replicate the expression at E14 into F14 to carry the values associated with the labels to this part of the sheet. Enter:

```
>E14:/R:F14:NNNNRR
```

By making the last two references relative, we used @LABEL to return the values in B1 and B2.

We now have a value associated with the BUY label that appears at E14. If, as above, we were to change the first quarter's earnings (at B8) from .22 to 1, E14 and F14 would change to SELL and 0, respectively. The distinguishable value associated with E14 can be used as a basis for comparison.

Now replicate E17...E20 into column F in order to associate values with the labels for each account. Enter:

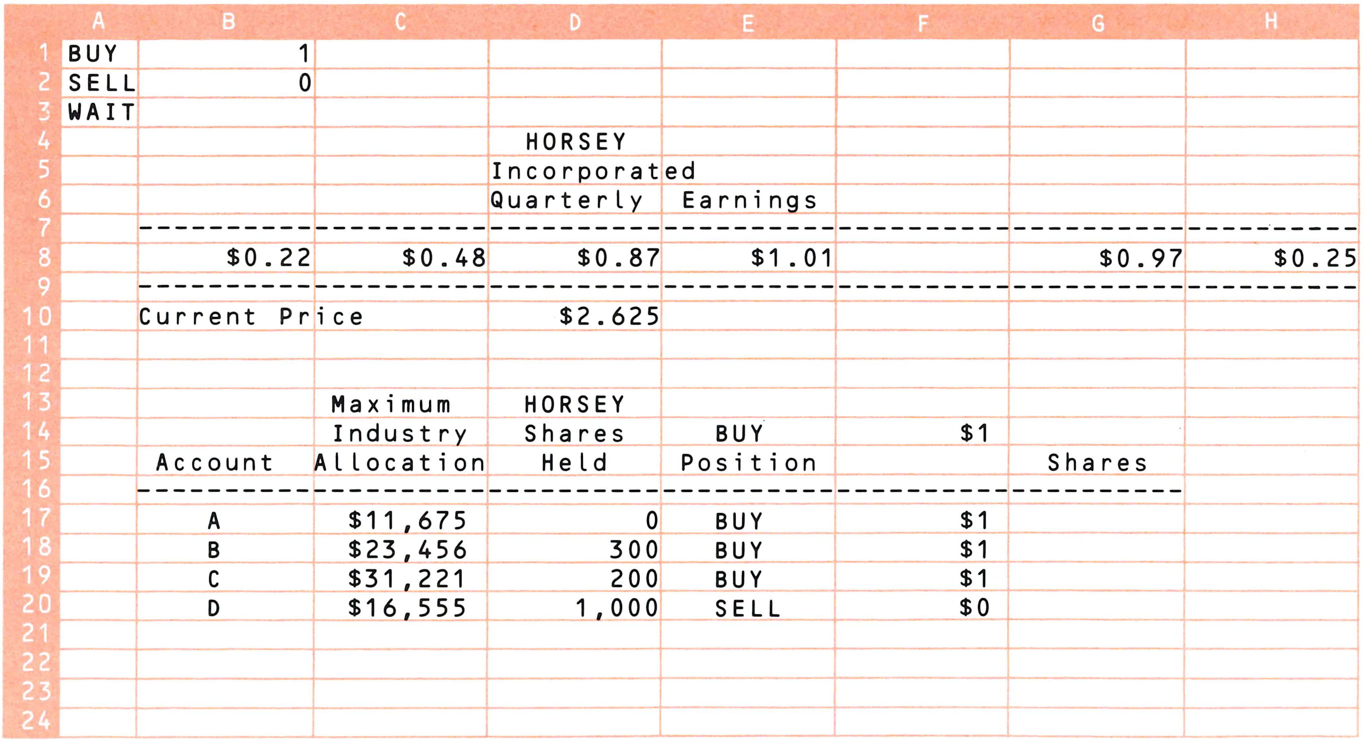
```
>E17:/R:F17:NNNRR
```

And:

```
>F17:/R:F18...F20:NRRNN
```

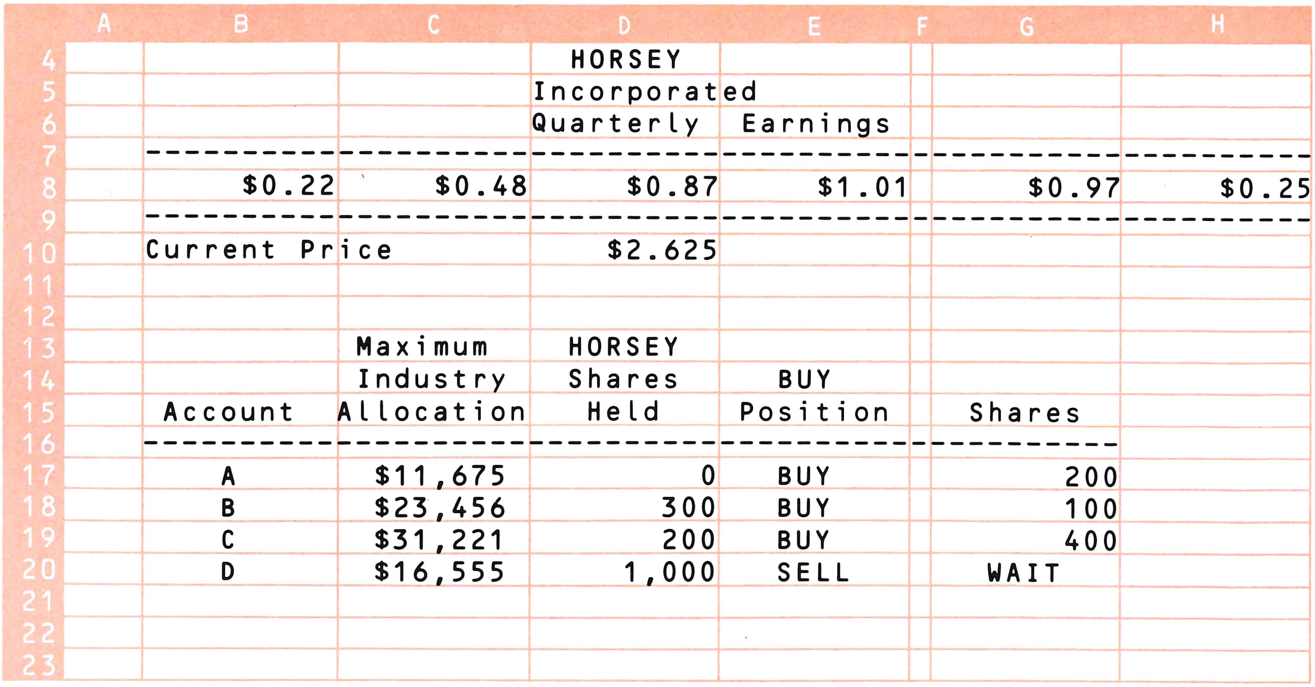
Your sheet should now look like the one shown in Figure 4.

We will use the ones and zeros associated with the BUY and SELL labels to calculate the number of shares (in hundreds) that should be purchased or sold for each account. But before we trade shares, we want to test whether both the average earnings and the number of shares held indicate the same action. Specifically, we want to compare the value in F14 with the ones in F17...F20. If we get one BUY and one SELL signal, that is, if the signals are mixed, we want to WAIT. For example, F14 indicates BUY, but F20, referring to Account D, indicates SELL, because the 1000 shares held exceed the 5% limit; therefore the decision to WAIT.



```
       A        B        C        D        E        F        G        H
 1 BUY             1
 2 SELL            0
 3 WAIT
 4                            HORSEY
 5                           Incorporated
 6                           Quarterly Earnings
 7          ---------------------------------------------------------------
 8             $0.22    $0.48    $0.87    $1.01             $0.97    $0.25
 9          ---------------------------------------------------------------
10          Current Price         $2.625
11
12
13                   Maximum  HORSEY
14                   Industry Shares      BUY          $1
15            Account Allocation Held    Position              Shares
16          ------------------------------------------------------
17              A     $11,675        0   BUY           $1
18              B     $23,456      300   BUY           $1
19              C     $31,221      200   BUY           $1
20              D     $16,555    1,000   SELL          $0
21
22
23
24
```

FIGURE 4. *Notice that those labels in column E that have been returned by the @LABEL function are centered in the column. Even though the program reads such labels as values, their display is controlled by formats that affect labels.*



	A	B	C	D	E	F	G	H
4				HORSEY				
5				Incorporated				
6				Quarterly	Earnings			
7		--------	--------	--------	--------	--------	--------	--------
8		$0.22	$0.48	$0.87	$1.01		$0.97	$0.25
9		--------	--------	--------	--------	--------	--------	--------
10		Current Price		$2.625				
11								
12								
13			Maximum	HORSEY				
14			Industry	Shares	BUY			
15		Account	Allocation	Held	Position		Shares	
16		--------	--------	--------	--------	--------	--------	
17		A	$11,675	0	BUY		200	
18		B	$23,456	300	BUY		100	
19		C	$31,221	200	BUY		400	
20		D	$16,555	1,000	SELL		WAIT	
21								
22								
23								

FIGURE 5. *This is the final form of the sheet discussed in the article. We have hidden column F from view by setting its width to one character and replicating the /AHY setting into the appropriate cells.*

If we get two BUY or two SELL signals, then we want to determine how many shares to trade at the current price, given the 5% limit. That means that we have to divide the 5% Maximum Industry Allocation by the Current Price. And since we already hold shares in HORSEY, we have to deduct the present holdings from the results.

Before we enter the appropriate expressions, we need to consider the second problem above.

Rounding to Hundreds

Since we want to avoid odd lot trades, we must round to the nearest hundred shares. This is where @ROUND comes in. The @ROUND function rounds the value specified as the first argument to the precision specified as the second argument. For example, if you wish to divide *value1* by *value2* and carry the operation to one decimal place, you would enter:

@ROUND(value1/value2,1)

If you omit the second argument, the function assumes it is 0, and rounds to an integer. Positive values, as in the example above, round to the right of the decimal place. (To round to the nearest penny, then, the expression would be *@ROUND(v1/v2,2)*.) Negative values round to the left of the decimal place. Thus, −1 rounds to the nearest ten, −2 rounds to the nearest hundred, and so on.

Finally, since the words BUY and SELL both imply positive actions, we want the results to appear as positive values. We can use the @ABS function to give us the absolute value of the shares to trade.

To accomplish all the above, enter:

```
>G17:/FI@IF(F14=F17,@ABS(@ROUND
  (.05*C17/D10,-2)-D17),@LABEL(A3))
```

Now replicate it:

```
>G17:/R:G18...G20:NRRNRN
```

We want to hide column F from the sheet, so move your cursor into that column and enter `/GCC1`. Then, to remove the > symbols, enter and replicate:

```
>F14:/AHY®®/R(A):F17...F20
```

That should make your sheet look like the one in Figure 5.

In closing, let us disclaim once again our stock selection method. It is quite intentionally simplistic and unorthodox, and we recommend that you not use it to make investment decisions. On the other hand, if you want to develop a report that uses @LABEL, redefine formats, or round values, keep a copy of this article on file; it will serve you well.

HAVE YOU MOVED?

If you have, let us know. To keep your issues of SATN coming to you regularly, we need to know your current address. Send a card or letter, indicating both your old address and your new one, to:

SATN Changes
P.O. Box 99
Newton Lower Falls, MA 02162

Remember to allow about 4 weeks for the change to take effect.

BOOK REVIEWS

Reviews published in SATN are the opinions of the reviewers, not of Software Arts, and may not be construed as official positions or endorsements by Software Arts.

Using VisiCalc: Getting Down to Business

by Carol Klitzner and Matthew J. Plociak, Jr. Wiley Press, a division of John Wiley & Sons, Inc., New York, 277 pp., $16.95 (softcover).

Mastering VisiCalc

by Douglas Hergert. Sybex Inc., Berkeley, 217 pp., $11.95 (softcover).

Reviewed by Barry Bayer

The Klitzner/Plociak book divides easily into two major sections. The first, which could be entitled "Learning VisiCalc," is about 170 pages long and is a reasonable beginner's textbook on the VisiCalc® program. It begins with information on how to insert a disk into a disk drive, continues with instructions on turning on the computer, and ends with Booleans, lookups (both @LOOKUP and @CHOOSE), and the like. Illustrations are given with respect to machine-specific operations on the Apple II, TRS-80 Model III, and IBM PC.

Practice exercises and review questions abound. The authors are generous with screen illustrations and include such highlighted rules of VisiCalc use as "Numbers in VisiCalc can't have commas." Anyone diligently working the exercises and reviews will gain a lot of VisiCalc experience, but will be particularly thankful when the concept of Replication is finally introduced on page 114.

A listing of the direct entries, function commands, and menu commands that have been discussed are presented at the end of each chapter. A typical entry tells us that "@LOOKUP(A1,B15...B23) looks up a value (first item in parentheses) in the table indicated by the range." The summary for the last chapter in the "textbook" section summarizes each of the functions and commands discussed in the first part of the book.

SUBMITTING ARTICLES TO SATN

Articles and pointers submitted to SATN are considered for acceptance based on their quality and interest to SATN readers.

We would prefer to see queries first. If you submit an article without querying us first, include a stamped, self-addressed envelope (or mailer for diskettes). If you do not enclose an SSAE or mailer, we cannot return your material. Prospective authors may send for the SATN Writer's Guide.

Address all queries, articles, or requests for the guide to:

SATN
Editorial Department
P.O. Box 99
Newton Lower Falls, MA 02162

The second, and weaker, part of the book is a 40-page chapter entitled "Getting Down to Business," which includes templates for a *potpourri* of business topics such as forecasting, cash flow, financial statements, financial ratios, economic order quantity, and breakeven analysis. A diskette containing the templates discussed in this section can be ordered from the publisher for $39.95; I did not have access to it. For those who do not purchase the diskette, a cell-by-cell "logic printout" of each file is included in an appendix.

A lot of accounting material is crammed into these pages, and it is not fast reading for someone who is unfamiliar with such matters. The textual discussion referring to the template printouts is sometimes difficult to follow, because no indication of column or row designation is given. You can get the information on a specific cell from the logic printouts, but that is not convenient.

The templates are mostly straightforward, but they don't, in my opinion, always use the best in VisiCalc technique. Variable factors tend to be built into a template, rather than placed in a separate "variable" area. To change a growth factor in a forecasting model, for example, it is necessary to edit the cell in the second month and replicate the new formula. Furthermore, although the CPAs I know who use the VisiCalc program make extensive use of rounding in their models (having been infected at certification with an incurable case of *nonfootaphobia*—an irrational fear of having someone notice that an array of figures does not crossfoot), rounding is not used in the sample financial templates, nor is there an entry for rounding in the index. The authors do provide three pages of good advice on "Designing Your Own VisiCalc Layouts," although it could have been profitably expanded into a full chapter.

The balance of the book consists of a 10-page overview of some of the features of the Advanced Version of the VisiCalc program and an appendix on "Transferring VisiCalc Data" using the DIF format. I agree with the authors' introductory comment that anyone who really likes the VisiCalc program "will love the advanced version," but I found no discussion of two of my favorite "advanced" features, variable column widths and the @ROUND function.

The first two-thirds of Hergert's book, *Mastering VisiCalc*, is a good, general-purpose introduction to the VisiCalc program, how to move around in the workspace and how to use the various commands and functions. The book does not begin on quite as elementary a level as Klitzner and Plociak's book does, assuming that the reader knows how to insert a disk into a drive and how to get the VisiCalc program running.

Hergert (author of *BASIC for Business* and other books) is apparently an experienced programmer, and his background shows. You certainly don't have to be a programmer to use the VisiCalc program, but a programmer's logical discipline certainly can't hurt. Chapter 5 is a 20-page section on how to plan a spreadsheet. It contains many well-founded practices, including the use of what Hergert calls "parameter tables," separate sections on the worksheet into which the "what if" type variables can be placed. Beginners will find this portion of the book useful, and even experts will enjoy the section on "Getting Along Without the Logical Functions."

The final third of *Mastering VisiCalc* goes into considerable detail about the structure of DIF files and how to use the VisiCalc program with your computer's "normal" operating environment. Hergert includes programs in BASIC that read, sort, and write DIF files. "A Brief Look at BASIC" is provided in Appendix B for those not familiar with BASIC. If you know BASIC but are unfamiliar with the disk file-handling techniques on either the Apple II, TRS-80 Model III, or IBM PC, Appendix A will fill that gap. It probably would be just as easy to transfer the BASIC programs to computers other than those mentioned above, because the programs are written in a "structured," easy-to-read format, with well-marked subroutines for disk access. I suspect that most readers of this review will not want to write a BASIC program to deal with DIF files, but any serious student of the VisiCalc program who ever unsuccessfully attempted to plow through Software Arts' Technical Note 18 (which was probably furnished with your VisiCalc manual) will enjoy working through the chapter.

Each book, then, has its place. *Using VisiCalc* should be purchased by the computer novice who seeks a formalized computer-lab-type setup where the review questions, exercises, and rules will be welcome. If I were using it in a classroom situation, however, I would direct the students to the section on Replicating long before they reach it in a page-by-page reading. The "Getting Down to Business" section has some useful information, but you may well decide to use the templates presented as a point of departure in your work, rather than using them "as is." *Mastering VisiCalc* is a good general introduction to the VisiCalc program and does not read as much like a textbook as *Using VisiCalc* does. For those who want to use DIF files in their BASIC programming and have been unable to figure out how to do it, *Mastering VisiCalc* is a must.

Barry Bayer is the author of "Visulating," a regular VisiCalc column appearing in Desktop Computing, *and co-author, with Joseph Sobel, of* Dynamics of VisiCalc.

POINTER: GIVING TABLES A DIFFERENT LOOK

How many times have you constructed an entire table, only to realize that it would be more effective if it were oriented in the opposite direction? In other words, the columns should be rows and vice versa.

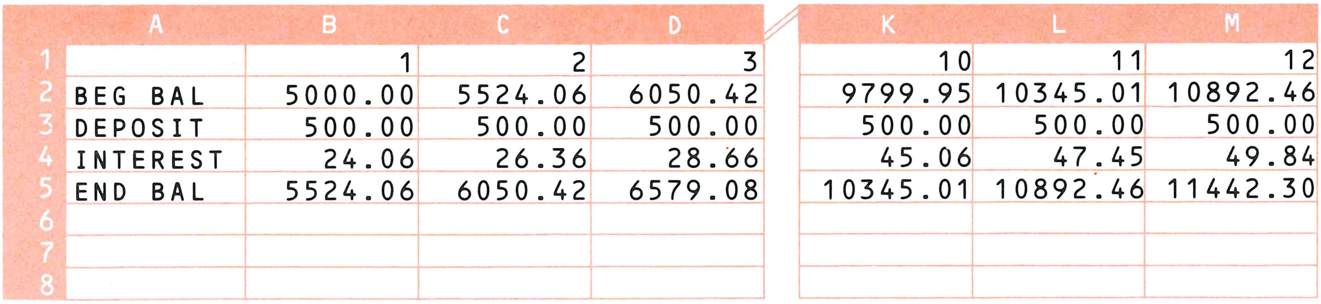
This can be done in VisiCalc® by using the DIF options /S#S and /S#L. We'll do it here using a table of monthly activity in a savings account.

The logical way to orient such a table is shown in Figure 6. With the normal (default) column width of 9, you can see from 4 to 8 columns at a time depending on which version of VisiCalc you are using. What if you want to see all 12 months at once? You can do that by changing the column widths, but the result can be hard to read since numbers may be truncated or converted to scientific notation if they are too long for the column width.

To set up the table for all twelve months as shown in Figure 6, first type **/GF$** for global formatting in dollars and cents. Next enter the labels in column A. Enter the period numbers in cells B1 to M1 with a local integer format (/FI). Then enter the following:

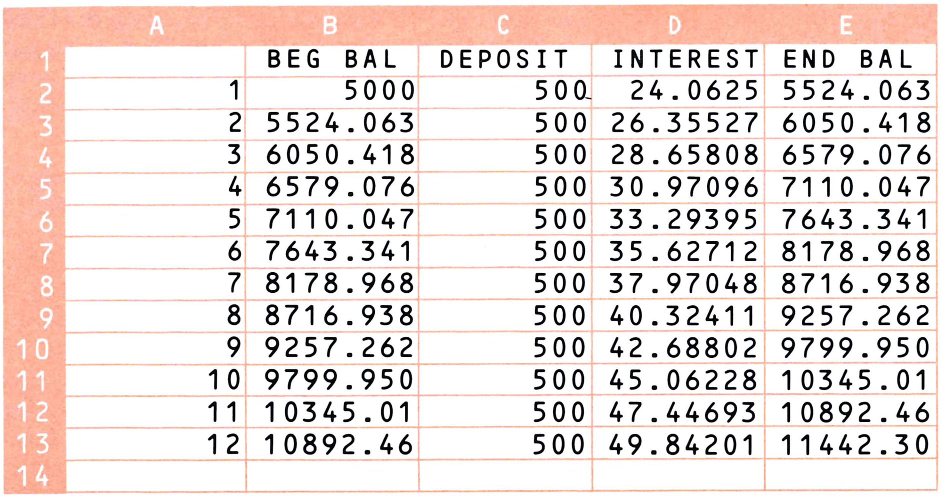
```
>B2:5000
>B3:500
>B4:(.0525/12)*(B2+B3)
>B5:@SUM(B2...B4)
>B3:/R...B5:C3...M3:RRRR
>C2:+B5®/R:D2...M2:R
```

Next, you should save the table in the usual fashion with **/SS**, because saving a DIF file saves only the numbers themselves, not the expressions from which some of the numbers are derived.



	A	B	C	D	K	L	M
1		1	2	3	10	11	12
2	BEG BAL	5000.00	5524.06	6050.42	9799.95	10345.01	10892.46
3	DEPOSIT	500.00	500.00	500.00	500.00	500.00	500.00
4	INTEREST	24.06	26.36	28.66	45.06	47.45	49.84
5	END BAL	5524.06	6050.42	6579.08	10345.01	10892.46	11442.30
6							
7							
8							

FIGURE 6 *This figure shows the activity by month in a savings account in which the beginning balance is $5000.00, deposits of $500.00 are made monthly, and 5-1/4% interest is compounded monthly.*



	A	B	C	D	E
1		BEG BAL	DEPOSIT	INTEREST	END BAL
2	1	5000	500	24.0625	5524.063
3	2	5524.063	500	26.35527	6050.418
4	3	6050.418	500	28.65808	6579.076
5	4	6579.076	500	30.97096	7110.047
6	5	7110.047	500	33.29395	7643.341
7	6	7643.341	500	35.62712	8178.968
8	7	8178.968	500	37.97048	8716.938
9	8	8716.938	500	40.32411	9257.262
10	9	9257.262	500	42.68802	9799.950
11	10	9799.950	500	45.06228	10345.01
12	11	10345.01	500	47.44693	10892.46
13	12	10892.46	500	49.84201	11442.30
14					

FIGURE 7. *This figure shows the table immediately after it has been loaded as a DIF file. Note the difference between formats in this figure and in Figure 6.*

Now we will save the table in a DIF file. First, position the cursor in the upper left corner of the data (in this case, cell A1) and type /S#S. You are then asked for a file name. In some versions of VisiCalc, the suffix .DIF is automatically appended to any DIF file name. In those versions in which .DIF is not automatically appended, you should use it as a suffix so that you can later identify the file as a DIF file. Once you have named the file, you are asked to specify the lower right of the sheet. In our example, it is M5, so enter M5. Next, you will be asked whether you want to save by row or by column. Typing R or RETURN will save by row and typing C will save by column. Whether you save by row or by column is immaterial, but it is important to remember which you choose.

To view the table in its reoriented form, first clear the worksheet, then load the file by typing /S#L. When you are asked whether you want to load by row or by column, respond with the opposite of the response you gave when asked how you wanted to save the worksheet. If you saved by column, load by row; if you saved by row, load by column. The table will then appear on the screen in its reoriented form (Figure 7).

The reoriented table may appear a bit strange, since a DIF file contains only numbers and labels. Expressions and formats that you used when you created the file are not saved as part of the DIF file. You may want to perform some cosmetic surgery by reformatting the table as you did when you first entered it. (An alternative is to enter the local integer format /FI in cell A2 and replicate it into cells A3 to A13 before loading the DIF file. After the file is loaded by /S#L, type /GF$ to put all other cells in dollars-and-cents format.)

Some words of warning:

- If you have saved a file using /S#S, trying to load it by /SL will result in an end-of-data flag (EOD) in cell A1. You must use /S#L to load the data.
- If you attempt to load a non-DIF file with /S#L, you will be asked if you want to load by row or by column. After you respond, the screen will remain blank.
- After a file has been saved by /S#S but before the screen has been cleared, expressions and formatting instructions are still available. If you forgot to save the file as a non-DIF file, you can still do so at this point.
- The location of the cursor when you enter the command /S#L determines the upper left corner of the data. Make sure you position the cursor in cell A1 (or wherever you want the data to begin) before loading the DIF file.

A useful resource for more complete information on the data interchange format is Donald H. Beil's *The DIF File: For Users of VisiCalc & Other Software*, published by Reston Publishing Company, Inc., Reston, Virginia.

BACK ISSUES AVAILABLE

Back issues of SATN are now available. The cost is $7.00 per copy for orders mailed within the USA. If your order is to be mailed outside the USA, add $2.50 for shipping and handling to the full price of the order.

To order back issues, please send your name and address, a list of the specific issues of SATN you want (by Volume and Issue numbers), and a check or money order payable to SATN to:

SATN Back Issues
P.O. Box 99
Newton Lower Falls, MA 02162